ASG
Software Solutions

ASG-DesignManager™ Enterprise Modeling

Version 1.4.3

Publication Number: DSR2200-143-ENT
Publication Date: July 1987

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 239.435.2200 Fax: 239.263.3692 Toll Free: 800.932.5536 (USA only)

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (239) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)
- Verify whether you received an ASG Service Pack for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack.

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578	239.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.222	49.89.45716.400	support.de@asg.com
Singapore	65.6332.2922	65.6337.7228	support.sg@asg.com
All other countries:	1.239.435.2200		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578		
Asia	65.6332.2922	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**Netherlands**	00.800.3354.3578
Denmark	00.800.9932.5536	**New Zealand**	00.800.9932.5536
France	00.800.3354.3578	**Singapore**	001.800.3354.3578
Germany	00.800.3354.3578	**South Korea**	001.800.9932.5536
Hong Kong	001.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Ireland	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536	**Thailand**	001.800.9932.5536
Japan/IDC	0061.800.9932.5536	**United Kingdom**	00.800.9932.5536
		All other countries	1.239.435.2200

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

Contents

Preface

This *ASG-DesignManager Enterprise Modeling* provides a top-down approach to data analysis. ASG-DesignManager (herein called DesignManager) is the interactive data and enterprise modeling system for logical and physical database design.

Allen Systems Group, Inc. (ASG) provides professional support to resolve any questions or concerns regarding the installation or use of any ASG product. Telephone technical support is available around the world, 24 hours a day, 7 days a week.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on any ASG product.

About this Publication

This publication consists of these chapters:

- Chapter 1, "Introduction to Enterprise Modeling," provides an introduction to DesignManger.
- Chapter 2, "Formulating Entities," provides a brief description of DesignManager's formulating entities.
- Chapter 3, "Associations Between Entities," provides a brief description of the associations between DesignManager's entities.
- Chapter 4, "Defining Entity Members," provides a brief description of DesignManager's entitiy members.
- Chapter 5, "Dependencies Formed From Entity Definitions," provides a brief description of dependencies formed from DesignManager's entity definitions.
- Chapter 6, "Syntax of Entity Member Definition Statements," provides a list of entity member definition statements when using DesignManager.

Publication Conventions

Allen Systems Group, Inc. uses these conventions in technical publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.
Vertical Separator Bar (|) with underline	Options available with the default value underlined (e.g., Y|N).

1 Introduction to Enterprise Modeling

Introduction

The Enterprise Modeling facility adopts the top-down approach to data analysis. To derive most benefit from reading of this publication, you need be familiar with these things:

- Manager Products and terminology
- Basic principles of data analysis
- The bottom-up approach to data analysis using DesignManager, which is described in the *ASG-DesignManager User's Guide*

Two Approaches to Data Analysis

The bottom-up approach is appropriate for well-defined existing activities for which detailed data is available. The best source of information for carrying out bottom-up analysis is usually user departments.

The top-down approach is especially useful for analyzing future or new application areas, where it is important to start with an overview of the application area before having to deal with a mass of detail, which might obscure the overall content and structure of the area being modeled.

For the top-down approach, you will find that you can often obtain the best overall picture from middle and upper management. They are usually in the best position to analyze the structure and overall functioning of an organization.

The Top-down Approach to Data Analysis

The top-down approach allows you to start with a skeleton outline of the enterprise being modeled and gradually add the detail as it becomes known.

Instead of data elements, your input to the DesignManager Enterprise Modeling facility consists of larger data objects called entities.

Entities

The Enterprise Modeling facility (selectable unit DSR-EM1O) provides for the definition of an additional DesignManager member type, ENTITY. In DesignManager, an entity is any object or concept that is of interest to an organization or activity area under investigation and about which information is kept or could be kept. Thus one of the first steps in adopting the Enterprise Modeling approach is to identify those objects, both physical and abstract, which a business or organization needs in order to operate. These are the entities with which Enterprise Modeling is concerned.

Examples of entities are:

EMPLOYEE	PATIENT	PRODUCT	SALES-AREA
DEPARTMENT	SUPPLIER	PROJECT	CLIENT
SALESMAN	CUSTOMER	PART	DOCTOR

What entities are relevant depends on the particular area under investigation.

The identification of relevant entities is best done incrementally. That is to say, additions and deletions can be made as more information becomes known or as errors are discovered. The main consideration at the initial stage of selecting prospective entities is that they should be data objects of interest in the area under investigation.

Note:

You should distinguish between an *entity* type and *instances* or *occurrences* of the entity type, for example: Entity type = EMPLOYEE Occurrences = J. Smith, A. Jones, etc.

DesignManager uses the term *entity* as shorthand for entity type and standardizes on *instance* instead of occurrence. Therefore, the example above becomes:

Entity = EMPLOYEE
Instances = J. Smith, A. Jones, etc.

Formulating Entities

Once you have identified entities, the next step is to determine relationships between them, for example:

BORROWER borrows BOOK
CUSTOMER orders PRODUCT
DOCTOR treats PATIENT
EMPLOYEE belongs to DEPARTMENT

In DesignManager, these relationships are called *associations*.

Note:

In DesignManager it is not necessary to state explicitly the nature of the association; it is merely necessary to specify that an association exists between one entity and another.

A further step in formulating entities is to determine the qualities and characteristics which describe the entities. In DesignManager, these qualities and characteristics are called *attributes* of the entity.

As an example, an entity BOOK could have these attributes:

BOOK-ISBN	BOOK-AUTHOR
BOOK-TITLE	BOOK-SUBJ
BOOK-CLASS-CODE	BOOK-PUB-DATE
BOOK-ACQ-DATE	

Using DesignManager for Enterprise Modeling

Having identified the entities, the associations, and perhaps some or all of the attributes, you then formally define the entities in the modeling dictionary as ENTITY members.

Note:

You can define an ENTITY member without any attributes or associations.

The entities are then added to the Workbench Design Area (WBDA) using the MERGE command. During processing of the MERGE command, they are converted to dependencies. The MERGE command may be used with both USERVIEW and ENTITY members. The collective term for USERVIEW and ENTITY members is *data-views*. Data-views may be collected together as *Viewsets* just before merging them in the WBDA. Viewsets can be used to group together data-views that belong to a particular area of the organization being modelled.

In specifying USERVIEW members, you explicitly define what dependencies are going to be produced in the WBDA; whereas, with ENTITY members, the dependencies are derived from the definitions by DesignManager.

The dependencies are then used to generate the relations and records of the conceptual schema.

The top-down and bottom-up approaches can be seen to differ chiefly in the gathering and input of data to the modeling dictionary rather than when operating on the contents of the WBDA. Thus, it is the input to the design process, rather than the process itself, which distinguishes between the top-down and the bottom-up approaches to data analysis using DesignManager.

Summary

The use of DesignManager requires these steps:

- Creating the ENTITY members and/or USERVIEW members
- Defining and entering them as members in the modeling dictionary
- Creating the composite view in the WBDA
- Generating the conceptual schema
- Producing a first-cut database design that is specific to a particular database management architecture, which may be relational, network, hierarchical, or inverted file

The last three steps are common to both bottom-up and top-down data analysis. Because these three aspects are covered in the *ASG-DesignManager User's Guide*, they will not receive detailed attention in this section.

2 Formulating Entities

Attributes of an Entity

An entity, once identified, has properties or characteristics that describe it. In DesignManager, these properties are called *attributes*. Possible attributes of an entity EMPLOYEE are employee number, name, address, salary, and skills. In DesignManager, attributes are GROUP and ITEM members of the modeling dictionary, for example:

EMPL-NO, EMPL-NAME, EMPL-SAL, EMPL-ADDR, EMPL-SKILL, CHILD-NAME, EMPL-PROJ, EMPL-DEPARTMENT

An employee will have one employee number, one name, one current salary, and one address. Therefore, EMPL-NO, EMPL-NAME, EMPL-SAL, and EMPL-ADDR are *one-attributes* of EMPLOYEE. One-attributes are specified using the ONE-ATTRIBUTES clause in an entity definition statement.

An Employee can have zero, one, or many skills. Therefore, EMPL-SKILL is a *multi-attribute* of EMPLOYEE. Multi-attributes are specified using MULTI-ATTRIBUTES clauses in entity definition statements.

Note:

Only one ONE-ATTRIBUTES clause needs to be specified for all of the one-attributes EMPL-NO, EMPL-NAME, EMPL-SAL, and EMPL-ADDR. However, a separate MULTI-ATTRIBUTES clause needs to be specified for each set of multi-attributes. For example, it is likely that one employee has many past salaries. To keep this information, you would probably want to record a salary date with each of the past salaries. Therefore, salary date and past salary would together qualify as a meaningful multi-attribute of the entity. SAL-DATE and PAST-SAL would appear together as a set in one MULTI-ATTRIBUTES clause.

This example illustrates the ONE-ATTRIBUTES and MULTI-ATTRIBUTES clauses:

```
ONE-ATTRIBUTES ARE EMPL-NAME, EMPL-SAL
MULTIATTRIBUTES ARE EMPL-SKILL
MULTIATTRIBUTES ARE SAL-DATE, PAST-SAL
```

In this example, if SAL-DATE and PAST-SAL did not appear together in a single MULTI-ATTRIBUTES clause, you would lose information, namely that SAL-DATE and PAST-SAL are a pair and cannot be separated in the context being described. It would be incorrect to include EMPL-SKILL in the same MULTI-ATTRIBUTES clause, because EMPL-SKILL, SAL-DATE, and PAST-SAL are not interdependent.

This is a representation of what might be an instance of the entity under discussion:

EMPLOYEE NUMBER	20105
EMPLOYEE NAME	SMITH, OLIVER
EMPLOYEE CURRENT SALARY	13500.65
EMPLOYEE PAST SALARY	1979 09350.00
	1982 10000.95
	1984 12769.00
EMPLOYEE SKILLS	COBOL
	ASSEMBLER
	JCL

The Identifier of an Entity

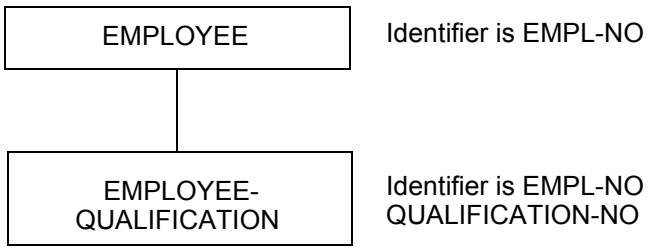
The data elements specified in the IDENTIFIER clause constitute the unique identifier of the entity being defined. Each value of the identifier determines exactly one specific instance of the entity. For example, a given employee number determines one and only one employee.

The identifier may consist of more than one data element and is specified in entity definition statements using the IDENTIFIER IS clause.

Dependent Entities

The identifier may contain the identifier of another entity. Consider a case where you have identified the entities EMPLOYEE and EMPLOYEE-QUALIFICATION. EMPLOYEE-QUALIFICATION has the attributes GRADE, WHERE-QUALIFIED, WHEN-QUALIFIED, EMPL-NO, and QUALIFICATION-NO together form the identifier of EMPLOYEE-QUALIFICATION (see Figure 1).

Figure 1. Dependent Entity

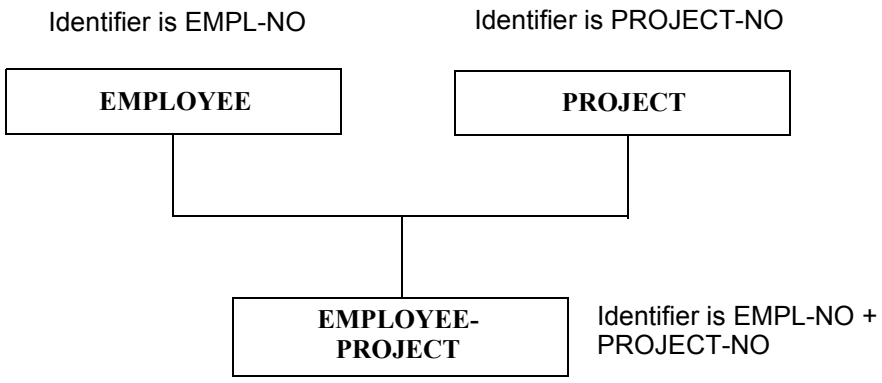


If QUALIFICATION-NO is not the identifier of another entity, EMPLOYEE-QUALIFICATION would be referred to as a *dependent* entity.

Intersection Entities

If the identifier contains the identifiers of two or more entities, then the entity being defined is an intersection entity. Its one-attributes are intersection data. In Figure 2, the identifier of EMPLOYEE-PROJECT is a combination of EMPL-NO and PROJECT-NO. EMPLOYEE-PROJECT is therefore an intersection entity; the attributes of EMPLOYEE-PROJECT are intersection data.

Figure 2. An Example of an Intersection Entity



The identifier of the entity EMPLOYEE-PROJECT is a combination EMPL-NO and PROJECT-NO. EMPLOYEE-PROJECT is therefore an *intersection entity*; the attributes of EMPLOYEE-PROJECT are *intersection data*.

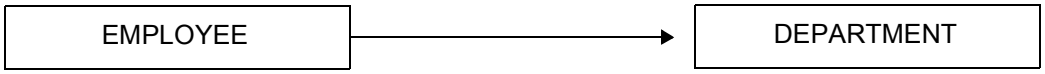
3 Associations Between Entities

Introduction

Associations between entities describe how the entities in the area being modeled relate to one another.

If an employee only ever works in one department, then there is a *one-association* between an employee and a department. Figure 3 shows a one-association between the entity EMPLOYEE and the entity DEPARTMENT:

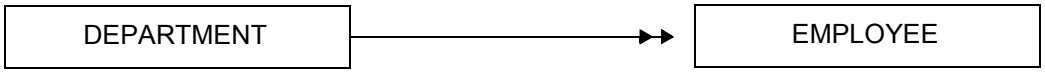
Figure 3. A One-Association between Employee and Department



One-associations from the entity being defined to another entity are specified using the ONE-ASSOCIATIONS clause.

Usually a department will contain more than one employee, as shown in Figure 4 :

Figure 4. A Multi-association between Department and Employee



This kind of association is called a multi-association in DesignManager.

Note:

The double headed arrow. MULTI-ASSOCIATIONS are specified using MULTI-ASSOCIATION clauses in entity member definitions.

Required Number of Clauses

Only one ONE-ASSOCIATION clause is required in an ENTITY definition statement. Assume that the entity EMPLOYEE is being defined. Assume that you want to establish a one-association between EMPLOYEE and DEPARTMENT, a one-association between EMPLOYEE and TELEPHONE, and a one-association between EMPLOYEE and PROJECT. Then this clause will suffice:

ONE-ASSOCIATIONS TO TELEPHONE, DEPARTMENT, PROJECT

In contrast, more than one MULTI-ASSOCIATION clause may be required in a definition to set up associations that are not interdependent. To illustrate, the clause:

MULTI-ASSOCIATION TO DEPARTMENT, PROJECT

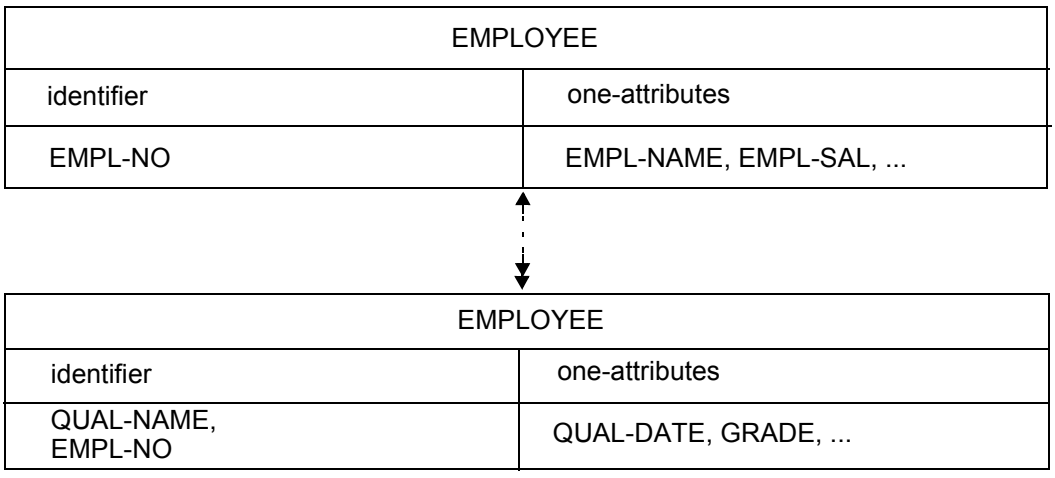
is not equivalent in meaning to the two clauses:

MULTI-ASSOCIATION TO DEPARTMENT
MULTI-ASSOCIATION TO PROJECT

The first clause reflects the fact that an employee does not belong to only one department, but works on various projects for various departments. When working on a given project, the EMPLOYEE is responsible to a given department. The two separate MULTI-ASSOCIATION clauses do not carry the same information. They suggest that an employee works on different projects in different departments, but projects are not associated to departments.

Hierarchical associations between entities are not directly specified by the user of the Enterprise Modeling facility. Instead they arise from situations where entities are defined as either dependent entities or intersection entities. A dependent entity is one whose identifier contains the identifier of another entity; an intersection entity is one whose identifier contains the identifier of two or more entities.

Figure 5. An Example of a Dependent Entity



Employee-Qualification has been defined with EMPL-NO as part of its identifier; it is therefore an example of a dependent entity. Figure 5 on page 10 illustrates a *hierarchical association.*

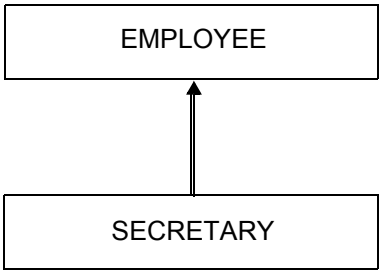
Hierarchical associations arise when and only when the identifier of an entity is contained in the identifier of another entity. In DesignManager, a hierarchical association should not be explicitly specified using the association clauses in ENTITY definition syntax, because hierarchical associations are implied as a consequence of the way identifiers are defined.

Subentities of an Entity

One further connection may be specified between entities; one entity may be specified as a *subentity* of another entity.

As an example, SECRETARY may be specified as a subentity of EMPLOYEE. This is shown in Figure 6 by the double-shafted arrow pointing from SECRETARY to EMPLOYEE:

Figure 6. An Example of a Subentity



Assume that the identifier of EMPLOYEE is EMPL-NO and assume that the identifier of SECRETARY is SEC-NO. Then the fact that we have specified that SECRETARY is a subentity of EMPLOYEE means that:

- Some instances of EMPLOYEE are also instances of Secretary.
- All instances of SECRETARY are also instances of Employee.
- Every attribute and association of EMPLOYEE is also an attribute or association of SECRETARY.

Further examples of subentities:

Entity: EMPLOYEE
Subentities: PROJECT-LEADER, PROGRAMMER, MANAGER

Entity: MEMBER-FIRM
Subentities: BUYER-FIRM, SELLER-FIRM

Entity: PROFESSOR
Subentities: ADVISING-PROFESSOR, TEACHING-PROFESSOR

Entity: PART
Subentities: MAJOR-PART, MINOR-PART

Entity: TELEPHONE
Subentities: EMPLOYEE-TELEPHONE, DEPARTMENT-TELEPHONE

Entity: VEHICLE
Subentities: LAND-VEHICLE, SEA-VEHICLE, AIR-VEHICLE

In the last example, the set of subentities do not overlap: a LAND-VEHICLE is always a LAND-VEHICLE and cannot normally be thought of as an AIR-VEHICLE.

However, some of the examples are cases where the subentities can overlap. For example, a professor may be an advising professor and a teaching professor.

One useful way of thinking about such subentities is to regard them as roles taken on by the given entity.

Entity Diagrams

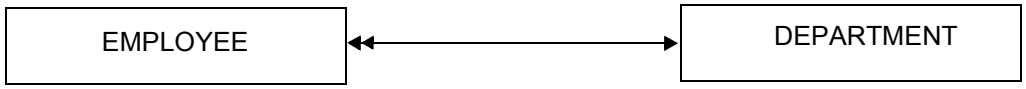
As an aid to using DesignManager for Enterprise Modeling, the use of entity diagrams is highly recommended.

Ideally, work on the diagram should start before any attempt is made to enter ENTITY members in the dictionary. This is because, in the course of drawing up the diagram, you may well discover that data objects that look like entities may turn out not to be entities. In the very act of drawing the diagram you may come across inconsistencies that you can eliminate before making any entries in the dictionary. In refining the diagram, you gradually build an overall picture of the area under investigation.

Start by listing the major data objects that you propose to treat as entities. Then begin to draw up the entity diagram.

An entity diagram consists of boxes connected by arrowed lines. Each box represents a separate entity. The arrowed lines represent the associations betwen the entities. Arrow-heads at the ends of the lines indicate the direction of each association. The type of arrow-head indicates the type of association: a double-headed arrow for a multi-association and a single-headed arrow for a one-association. A single line arrowed at both ends may be used to represent two associations, one in each direction. The entity diagram (see Figure 7) shows two associations, namely a multi-association from DEPARTMENT to EMPLOYEE and a one-association from EMPLOYEE to DEPARTMENT:

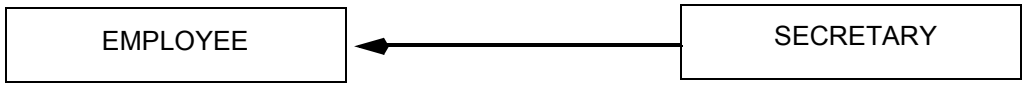
Figure 7. Part of an Entity Diagram Showing Two Associations, One in Each Direction



You may add to the entity diagram any information that clarifies it, such as the nature of an association. However, you should avoid excessive detail; for example, attributes or identifiers of entities should not appear on an entity diagram.

On entity diagrams, a double-shafted arrow can be used to represent a subentity relationship with the arrow pointing away from the subentity, as in Figure 8.

Figure 8. Part of an Entity Diagram Showing a Subentity Relationship



Each box should be labelled with the name of the proposed entity, and, ASG recommends that you use the full names of proposed entities rather than abbreviations.

Once you are satisfied that you have identified at least the major entities and the associations between them, you can then proceed to enter ENTITY members in the modeling dictionary.

A complete entity diagram shows all the entities identified so far in the area under investigation. It can be refined as more information is gathered before, during, and after successive MERGE DESIGN, REPORT, and PLOT cycles using DesignManager.

Entity diagrams can be used for:

- Starting out with an overall picture, so that missing or redundant entities can be included or eliminated at an early stage
- Helping with the naming of ENTITY, USERVIEW, and VIEWSET members
- As an aid to interpreting subsequent output from the MERGE, DESIGN, LIST, REPORT, and PLOT commands

Possible combinations of one-associations and multi-associations can be pictured in entity diagrams by bidirectional arrows. Figure 9, Figure 10, and Figure 11 show examples of different combinations:

Figure 9. A One-association in Both Directions

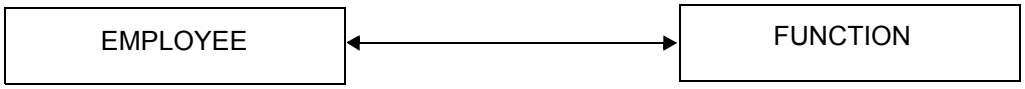


Figure 10. A Multi-association in Both Directions

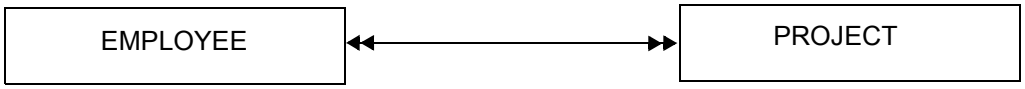
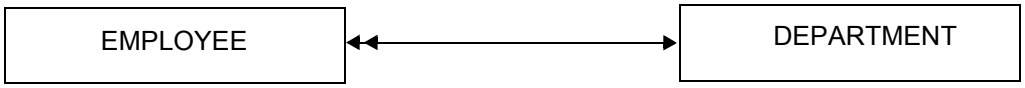


Figure 11. A Multi-association Exists from DEPARTMENT to EMPLOYEE, and a One-association from EMPLOYEE to DEPARTMENT



4 Defining Entity Members

Dependencies

This section describes briefly how dependencies are derived from individual clauses in ENTITY member definitions. The following is an example definition, which would be entered as a member of the modeling dictionary either via an ADD command or through an update buffer.

Entity definition:

```
ADD EMPLOYEE;
ENTITY
IDENTIFIER IS EMPL-NO
ONE-ATTRIBUTES ARE EMPL-NAME, EMPL-ADDR, NO-OF-DEPEND
MULTI-ATTRIBUTES ARE DEPEND-NAME
MULTI-ATTIBUTES ARE SAL-DATE, PAST-SAL
ONE-ASSOCIATIONS TO DEPARTMENT, OFFICE
MULTI-ASSOCIATION TO PROJECT
MULTI-ASSOCIATION TO SKILL
MULTI-ASSOCIATION TO EDUC-INST
MULTI-ASSOCIATION TO PAST-EMPLOYER
MULTI-ASSOCIATION TO POSITION-HELD
MULTI-ASSOCIATION TO PAST-EMPLOYER, POSITION-HELD
SUB-ENTITIES ARE MANAGER, ENGINEER, CLERK
;
```

Dependencies are generated from an ENTITY member when the member is MERGED:

- A functional dependency (FD) is generated for each of the data elements in the ONE-ATTRIBUTES clause between the identifier and data element.
- An FD is generated for each of the entities in the ONE-ASSOCIATIONS clause between the identifier and the identifier of each entity named in the clause.
- One multivalued dependency (MVD) is generated, for each MULTI-ATTRIBUTES clause between the identifier and all of the data elements named in the clause.
- One MVD is generated, for each MULTI-ASSOCIATION clause between the identifier and all of the identifiers of the entities named in the clause.

Entity Member Names

Entity member names must conform to the rules for naming dictionary members as set out in the *ASG-DesignManager User's Guide*.

In particular, users of the Enterprise Modeling facility should observe the indicated constraints with respect to entity and data element names; otherwise naming conflicts may arise in the Workbench Design Area (WBDA) during processing of the MERGE command.

The problems associated with the avoidance of homonyms and synonyms are the same as those encountered in naming data elements. These problems can be dealt with in a similar fashion to that described for data elements in the *ASG-DesignManager User's Guide.*

A homonym could be generated in the WBDA if the dictionary name given to an entity or data element were to start with the default character.

Particular care should be taken to avoid synonymous entity names such as CLIENT and CUSTOMER.

Care must also be taken not to give different entities the same identifier.

It is helpful to maintain in the dictionary a company-wide list of standard keywords with standard abbreviations. It should then be standard practice to use the full keyword for naming entities and the abbreviations for naming data elements. In general, the best safeguard against naming conflicts is to maintain strict naming standards supported by the facilities offered by the modeling dictionary.

The IDENTIFIER Clause

The data elements specified in the IDENTIFIER clause constitute the unique identifier of the entity being defined. Each value of the identifier determines exactly one specific instance of the entity. For example, a given employee number determines one and only one employee.

No two entities should be given the same identifier, because the software does not do any internal validation to ensure that identifiers are different.

However, the user's failure to distinguish between identifiers can lead to undesirable results. For example, the action of a subsequent DESIGN command on two entities with the same identifier would result in the formation of a single relation containing the attributes of both entities.

In addition, if either entity were to be named in the ONE-ASSOCIATIONS clause of the other, the action of a subsequent MERGE command would include the generation of a dependency with identical left and right hand sides. Such a dependency would not be added to the Workbench Design Area. The same applies in cases where either such entity is the only entity named in the MULTI-ASSOCIATION clause of the other.

The ONE-ATTRIBUTES and MULTI-ATTRIBUTES Clauses

The ONE-ATTRIBUTES Clause

Each data element named in the ONE-ATTRIBUTES clause is a functionally dependent attribute of the entity being defined. That is, it is functionally dependent on the entity's identifier. Each instance of the entity, and therefore each value of any identifier specified for the entity, determines exactly one value of the attribute.

Note:

Only one ONE-ATTRIBUTES clause is required in an entity definition. This is because the clause:

```
ONE-ATTRIBUTES ARE EMPL-NAME, EMPL-ADDR, EMPL-SAL
```

is converted during processing of the MERGE command into three separate functional dependencies, from the identifier of the entity being defined to each of EMPL-NAME, EMPL-ADDR, and EMPL-SAL, as follows (where EMPL-NO is the identifier of the entity being defined):

```
EMPL-NO -----> EMPL-NAME
EMPL-NO -----> EMPL-ADDR
EMPL-NO -----> EMPL-SAL
```

The MULTI-ATTRIBUTES Clause

The MULTI-ATTRIBUTES clause, if present in the entity's definition, contains a set of one or more data elements. This set of attributes is multiply determined by the value of the identifier. Each value of the identifier of the entity being defined determines zero, one, or many values of the combined set of attributes.

There can be more than one MULTI-ATTRIBUTES clause in a definition. Each clause will be converted by the MERGE command into a multivalued dependency (MVD).

Note:

For this reason, the clause:

MULTI-ATTRIBUTES ARE SALDATE, PAST-SAL

is *not* equivalent in meaning to these two clauses:

MULTI-ATTRIBUTES ARE SAL-DATE
MULTI-ATTRIBUTES ARE PAST-SAL

The first clause would generate this multivalued dependency:

EMPL-NO -----> SAL-DATE, PAST-SAL

The other two clauses would generate these dependencies:

EMPL-NO -----> SALDATE
EMPL-NO -----> PAST-SAL

The ONE-ASSOCIATIONS and the MULTI-ASSOCIATION Clauses

The ONE-ASSOCIATIONS Clause

For each entity named in any ONE-ASSOCIATIONS clause, a one-association holds from the entity being defined to the named entity. The identifier of the named entity is functionally dependent on the identifier of the entity being defined. A functional dependency (FD) will be generated during processing of the MERGE command from the identifier of the entity being defined to the identifier of the named entity.

Only one ONE-ASSOCIATIONS clause is required in an entity definition, for the same reason that only one ONE-ATTRIBUTES clause is required. This is because the clause:

ONE-ASSOCIATIONS TO DEPARTMENT, TELEPHONE

means that a one-association holds between an employee and a particular department, and that a one-association also holds between an employee and a particular telephone number. In other words, an employee works for one department and an employee has one telephone number.

Thus, assuming that EMPL-NO is the identifier of the entity being defined, DEPARTMENT-NO the identifier of the entity DEPARTMENT, and TELEPHONE-NO is the identifier of the entity TELEPHONE, then this clause will generate these functional dependencies:

EMPL-NO -----> DEPARTMENT-NO
EMPL-NO -----> TELEPHONE-NO

The MULTI-ASSOCIATION Clause

There can be more than one MULTI-ASSOCIATION clause in an entity's definition.

For each clause present, a multi-association holds from the entity being defined to the set of entities named in the clause. A multivalued dependency (MVD) is generated during processing of the MERGE command from the identifier of the entity being defined to the set of the identifiers of all the entities named in the clause. This set of data elements is multiply determined by the identifier of the entity being defined.

Thus, if EMPLR-NO is the identifier of the entity PAST-EMPLOYER, POSTHELD the identifier of the entity POSITION-HELD, and EMPL-NO the identifier of the entity being defined, then the clause:

```
MULTI-ASSOCIATION TO PAST-EMPLOYER, POSITION-HELD
```

means that the set of data elements EMPLR-NO and POS-HELD is multiply dependent on the data element EMPL-NO.

The MULTI-ASSOCIATION clause is repeatable for the same reason that the MULTI-ATTRIBUTE clause is repeatable.

The clause:

```
MULTI-ASSOCIATION TO PAST-EMPLOYER, POSITION-HELD
```

is not equivalent in meaning to the two clauses:

```
MULTI-ASSOCIATION TO PAST-EMPLOYER
MULTI-ASSOCIATION TO POSITION-HELD
```

This MVD would be generated from the first clause:

```
EMPL-NO ---->> EMPLR-NO, POS -HELD
```

The other two clauses would be converted to two separate MVDs:

```
EMPL-NO ---->> EMPLR-NO
EMPL-NO ---->> POS-HELD
```

The SUB-ENTITIES Clause

Only one SUB-ENTITIES clause need be specified in an entity definition. Each entity named in a SUB-ENTITIES clause is a subentity of the entity being defined. The identifier of each named entity is a subcategory of the identifier of the entity being defined. The identifier of each subentity must be different from the identifier of the entity being defined.

During processing of the MERGE command, a *domain dependency* (DD) is generated from the identifier of the named entity to the identifier of the entity being defined.

Thus, assuming that EMPL-NO is the identifier of the entity being defined, and CLER-EMPL-NO is the identifier of an entity CLERK, and further, that MGR-EMPL-NO is the identifier of an entity MANAGER, then the clause:

SUB-ENTITIES ARE MANAGER, CLERK

will generate these domain dependencies:

CLER-EMPL-NO = = = = => EMPL-NO
MGR-EMPL-NO = = = = => EMPL-NO

Common Clauses

Common clauses can be present in any type of member definition. A given common clause can appear only once in a definition.

In ENTITY definitions, common clauses can appear in any order and in any position between the ENTITY keyword and the statement terminator. A common clause cannot appear within another common clause or within any of the other optional clauses.

Remarks on Entity Definitions

Note these points with regard to entity definitions:

- The clauses are optional, which means that you can begin by identifying and adding an ENTITY by name only and gradually add information about its attributes and associations as the details become evident.
- The MULTI-ATTRIBUTES clause and the MULTI-ASSOCIATION clause are repeatable; however, there need only be one ONE-ATTRIBUTES clause, one ONE-ASSOCIATIONS, and one SUB-ENTITIES clause in the definition.
- Attributes referenced in the definition are data-elements; that is, they are ITEM or GROUP members of the modeling dictionary.
- The members named in ONE-ASSOCIATIONS, MULTI-ASSOCIATION, and SUB-ENTITIES clauses are ENTITY members.
- If an attribute referenced in the definition of an ENTITY member has no Data Entries record in the modeling dictionary then when the member is encoded, a DUMMY member of type ITEM containing used-by pointers is created in the modeling dictionary.
- If an ENTITY referenced in the definition of an ENTITY member has no Data Entries record in the modeling dictionary, then when the member is encoded, a DUMMY member of type ENTITY containing used-by pointers is created in the modeling dictionary.
- If a member referenced in a clause is an existing dummy member of type VIEWSET then when the current ENTITY is encoded, the member type of the dummy is changed to ENTITY. If the named member is an existing dummy ENTITY, then the type of the dummy remains unchanged.
- ENTITY members, like USERVIEW members, can be contained in VIEWSETs for ease of specification in the MERGE command.
- When naming entities in member definitions, take care to avoid homonyms and synonyms. The problems with naming entities are the same as those described for naming data-elements and USERVIEW members. Take particular care not to give the same name to an entity and a data-element.

ENTITY members, like ITEM members, GROUP members, and USERVIEW members, are defined and entered as members of the modeling dictionary. Therefore, once entered in the dictionary, they can be reported on, updated, and can be the subject of interrogation commands like any other dictionary member.

Common clauses may be contained in the member definition, subject to the same rules that apply to other member type definitions.

5 Dependencies Formed From Entity Definitions

Rules For Generating Dependencies

Clauses in ENTITY member definition statements are optional. Therefore the following rules only apply to clauses that are present in a given definition.

Defaults

- When you define an entity in the modeling dictionary there is no need to define a ONE-ATTRIBUTES clause straight away. Recall also that the IDENTIFIER does not need to be specified when the member is defined. This means that it is possible to have a member defined without a ONE-ATTRIBUTES clause or an IDENTIFIER clause. However, the action of the MERGE command always results in the creation of at least one functional dependency (FD) from the identifier to a one-attribute, whether or not any clauses are present in the definition. To cover cases where one-attributes or identifiers have not been specified, these defaults are supplied:

  — If no IDENTIFIER clause is present, a default identifier is supplied. This consists of the entity's name prefixed by an explanation point (!). For an entity named EMPLOYEE, the default identifier would be EMPLOYEE. This prefix character can be tailored. (Refer to the LHSPRE parameter in the LOPT1 macro in the *ASG-Manager Products Installation in OS Environments* or the *ASG-Manager Products Installation in DOS Environments* publications.)

  — If no ONE-ATTRIBUTES clause is present in the definition, a default one-attribute is generated. This consists of the entity's name, prefixed by the at sign（@）. In the case of the EMPLOYEE entity, the default one-attribute would be @EMPLOYEE. This prefix character can be tailored (refer to the RHSPRE parameter in the LOPTl macro in the appropriate publication).

  Therefore, if no IDENTIFIER or ONE-ATTRIBUTES clause were present in the definition of the entity EMPLOYEE, then this FD would be generated and placed in the Workbench Design Area (WBDA) during processing of the MERGE command:

  !EMPLOYEE -----> @EMPLOYEE

  Note:

  When such defaults need to be generated, if the entity name is longer than 31 characters, then DesignManager will reject the entity for the current MERGE command.

- A separate FD is generated for each data element specified in the ONE-ATTRIBUTES clause. The identifier of the entity is the left-hand side (LHS) of the dependency and the data element is the right-hand side (RHS).
- A separate multivalued dependency (MVD) is generated for each MULTI-ATTRIBUTES clause present. The LHS is the entity's identifier and the RHS is the set of data elements specified in the MULTI-ATTRIBUTES clause. If no identifier is present in the member's definition, the default identifier is used. If no MULTI-ATTRIBUTES clause is present, no default MVD is generated.
- An FD is generated for each entity named in a ONE-ASSOCIATIONS clause. The LHS of the FD is formed by the identifier of the entity being defined. The RHS of the FD is formed by the identifier of the entity named in the ONE-ASSOCIATIONS clause.
- If no identifier is specified for the entity being defined, the default is used. If no identifier has been specified for the entity being named in the clause, then the default identifier for it is used. This also applies if the named entity is a dummy. If no ONE-ASSOCIATIONS clause is present, then no default FD is generated.
- For each MULTI-ASSOCIATION clause, a separate MVD is generated. The dependency is from the identifier of the entity being defined to the identifiers of the entities named in the clause. If no identifier is specified for the entity being defined, the default identifier is used. If no identifier has been specified for an entity referenced in the MULTI-ASSOCIATION clause, then for each such entity, the default identifier is used to form the MVD. This also applies if the referenced entity is a dummy ENTITY member. If no MULTI-ASSOCIATION clause is present, no default MVD is generated.
- For each entity specified in the SUB-ENTITIES clause, a domain dependency (DD) is generated and added to the (WBDA). The dependency is from the identifier of the entity named in the SUB-ENTITIES clause to the identifier of the given entity.
- As in the case of ONE-ASSOCIATIONS and MULTI-ASSOCIATIONS, default identifiers are supplied as required.
- If an entity E has been specified in the ONE-ASSOCIATIONS, MULTI-ASSOCIATION or SUB-ENTITIES clauses of a given entity's data definition, and if E is an encoded member of the modeling dictionary, then the identifier of E is available for forming dependencies during processing of the MERGE command, even if E is not explicitly one of the entities being merged.

Example Entity Definitions

```
ADD DEPARTMENT;
ENTITY
IDENTIFIER IS DEPARTMENT-NO
ONE-ATTRIBUTES ARE DEPARTMENT-NAME, BLDG-NO
MULTI-ASSOCIATION TO PROJECT
MULTI-ASSOCIATION TO OFFICE
;

ADD OFFICE;
ENTITY
IDENTIFIER IS OFFICE-NO
ONE-ATTRIBUTES ARE BLDG-NO, OFFICE-TEL, NO-OF-EMP
;

ADD FUNCTION;
ENTITY
IDENTIFIER IS DEPARTMENT-NO, OFFICE-NO
ONE-ATTRIBUTES ARE MAJ-FUNCT
;

ADD EMPLOYEE;
ENTITY
IDENTIFIER IS EMPL-NO
ONE-ATTRIBUTES ARE EMPL-NAME, EMPL-ADDR, NO-OF-DEPEND
MULTI-ATTRIBUTES ARE DEPEND-NAME
MULTI-ATTRIBUTES ARE SAL-DATE, PAST-SAL
ONE-ASSOCIATIONS TO DEPARTMENT, OFFICE
MULTI-ASSOCIATION TO PROJECT
MULTI-ASSOCIATION TO SKILL
MULTI-ASSOCIATION TO EDUC-INST
MULTI-ASSOCIATION TO PAST-EMPLOYER
MULTI-ASSOCIATION TO POS~ON-HELD
MULTI-ASSOCIATION TO PAST-EMPLOYER, POSITION-HELD
SUB-ENTITIES ARE MANAGER, ENGINEER, CLERK
;

ADD EMPLOYEE-NEXT-OF-KIN;
ENTITY
IDENTIFIER IS EMPL-NO, DEPEND-NAME
ONE-ATTRIBUTES ARE DEPEND-CATEG
;

ADD MANAGER;
ENTITY
IDENTIFIER IS MGWEMPL-NO
ONE-ASSOCIATIONS TO DEPARTMENT
COMMENT 'MANAGER IS A SUBENTITY OF EMPLOYEE'
;
```

```
ADD ENGINEER;
ENTITY
COMMENT 'ENGINEER IS A SUBENTITY OF EMPLOYEE'
;

ADD CLERK;
ENTITY
IDENTIFIER IS CLER-EMPL-NO
COMMENT 'CLERK IS A SUBENTITY OF EMPLOYEE'
;

ADD SKILL;
      ENTITY
      ONE-ATTRIBUTES ARE SKILL-NAME
      MULTI-ASSOCIATION TO EMPLOYEE
      ;

ADD PAST-EMPLOYER;
ENTITY
IDENTIFIER IS EMPLR-NAME
ONE-ATTRIBUTES ARE EMPLR-ADDR, EMPLR-TYPE
;

ADD POSITION-HELD;
ENTITY
IDENTIFIER IS JOB-CATEG
;
```

The following VIEWSET definition gathers together 11 of the 12 ENTITY member definitions for merging in the Workbench Design Area. A VIEWSET member may include USERVIEW members as well as ENTITY members:

```
      ADD ORGANIZATION;
      VIEWSET
      CONTAINS  DEPARTMENT, OFFICE, FUNCTION, EMPLOYEE,
                     MANAGER, ENGINEER, PROJECT, SKILL, PAST-EMPLOYER,
                     POSITION-HELD, EMPLOYEE-NEXT-OF-KIN
      ;
```

Note:

The CLERK entity has not been included in the VIEWSET. Nevertheless, the identifier CLER-EMPL-NO is avaliable for use as required in the RHS of a functional or multivalued dependency and in the left-hand-side of a domain dependency.

Dependencies Generated from Example Definitions

The command:

MERGE MEMBERS ORGANIZATION;

causes these dependencies to be generated and placed in the Workbench Design Area (WBDA):

For entity DEPARTMENT:

DEPARTMENT-NO -----> DEPARTMENT-NAME
DEPARTMENT-NO -----> BLDG-NO
DEPARTMENT-NO ---->> PROJ-NO
DEPARTMENT-NO ---->> OFFICE-NO

For entity OFFICE:

OFFICE-NO -----> BLDG-NO
OFFICE-NO -----> OFFICE-TEL
OFFICE-NO -----> NO-OF-EMP

For entity FUNCTION:

DEPARTMENT-NO, OFFICE-NO -----> MAJ-FUNCT

For entity EMPLOYEE:

EMPL-NO -----> EMPL-NAME
EMPL-NO -----> EMPL-ADDR
EMPL-NO -----> NO-OF-DEPEND
EMPL-NO ---->> DEPEND-NAME
EMPL-NO ---->> SAL-DATE, PAST-SAL
EMPL-NO -----> DEPARTMENT-NO
EMPL-NO -----> OFFICE-NO
EMPL-NO ---->> PROJ-NO
EMPL-NO ---->> SKILL
EMPL-NO ---->> !EDUC-INST
EMPL-NO ---->> EMPLR-NAME
EMPL-NO ---->> JOB -CATEG
EMPL-NO ---->> EMPLR-NAME,
MGR-EMPL-NO = = = = => EMPL-NO
ENGINEER = = = = => EMPL-NO
CLER-EMPL-NO = = = = => EMPL-NO

For entity EMPLOYEE-NEXT-OF-KIN:

EMPL-NO, DEPEND-NAME -----> DEPEND-CATEG

For entity MANAGER:

MGR-EMPL-NO -----> @MANAGER
MGR-EMPL-NO -----> DEPARTMENT-NO

Note:

Default one-attribute generated.

For entity ENGINEER:

!ENGINEER ----->@ENGINEER

Note:

Default identifier and default one-attribute generated.

For entity PROJECT:

PROJ-NO -----> PROJ-NAME
PROJ-NO -----> PROJ-TYPE
PROJ-NO -----> DEPARTMENT-NO
PROJ-NO ---->> EMPL-NO

For entity SKILL:

!SKILL -----> SKILL-NAME
!SKILL ---->> EMPL-NO

Note:

Default identifier generated.

For entity PAST-EMPLOYER:

EMPLR-NAME - - - - - > EMPLR-ADDR
EMPLR-NAME - - - - - > EMPLR-TYPE

For entity POSITION-HELD:

JOB-CATEG - - - - - > @POSITION-HELD

Note:

Default one-attribute generated.

Note:

Default FDs are not generated for the CLERK or EDUC-INST entities because they are not specified, either directly or indirectly, in the MERGE command. However, the identifier CLER-EMPL-NO of the CLERK entity was available for use as the left-hand side in a domain dependency to EMPL-NO.

The Rest of the Design Process

Once entities are converted to dependencies and added to the Workbench Design Area (WBDA), the remainder of the design procedure is the same as for the bottom-up approach as described in the *ASG-DesignManager User's Guide*.

Following successful processing of the MERGE command, it is a useful step to obtain an Intersecting Data Elements Report. Further reports available at this stage include:

- The Data Elements Usage Analysis Report
- The Entity Report
- The Relational Schema Report (following a DESIGN command)

If you have the optional User Printer Graphics facility (selectable unit DSR-UD31) installed, various outputs from the PLOT command are available. In particular, the PLOT NETWORK SCHEMA command can be helpful in the top-down approach because the network produced is similar to a set of entities. You can therefore use it as an iterative control mechanism for modifying and re-merging entities and for refining an entity diagram.

Notes on the EMPLOYEE Entity

Entity definition:

```
ADD EMPLOYEE;
ENTITY
IDENTIFIER IS EMPL-NO
ONE-ATTRIBUTES ARE EMPL-NAME, EMPL-ADDR, NO-OF-DEPEND
MULTI-ATTRIBUTES ARE DEPEND-NAME
MULTI-ATTRIBUTES ARE SAL-DATE, PAST-SAL
ONE-ASSOCIATIONS TO DEPARTMENT, OFFICE
MULTI-ASSOCIATION TO PROJECT
MULTI-ASSOCIATION TO SKILL
MULTI-ASSOCIATION TO EDUC-INST
MULTI-ASSOCIATION TO PAST-EMPLOYER
MULTI-ASSOCIATION TO POSITION-HELD
MULTI-ASSOCIATION TO PAST-EMPLOYER, POSITION-HELD
SUBENTITIES ARE MANAGER, ENGINEER, CLERK
;
```

Dependencies are generated from the above definition as follows:

```
EMPL-NO -----> EMPL-NAME
EMPL-NO -----> EMPL-ADDR
EMPL-NO -----> NO-OF-DEPEND
EMPL-NO ---->> DEPEND-NAME
EMPL-NO ---->> SAL-DATE , PAST-SAL
EMPL-NO -----> DEPARTMENT-NO
EMPL-NO -----> OFFICE-NO
EMPL-NO ---->> PROJ-NO
EMPL-NO ---->> !SKILL
EMPL-NO ---->> !EDUC-INST
EMPL-NO ---->> EMPLR-NAME
EMPL-NO ---->> JOB-CATEG
EMPL-NO ---->> EMPLR-NAME, JOB-CATEG
MGR-EMPL-NO =====> EMPL-NO
!ENGINEER =====> EMPL-NO
 CLER-EMPL-NO =====> EMPL-NO
```

Although the entity CLERK is not included in the VIEWSET that is merged, its identifier is available and is used in a domain dependency generated from the SUB-ENTITIES clause of the EMPLOYEE entity.

A default identifier for the dummy entity EDUC-INST has been generated and used in the EMPLOYEE dependencies. The same has been done for the entities SKILL and ENGINEER, which have no identifier specified.

Two MULTI-ATTRIBUTES clauses are needed for the EMPLOYEE entity in order to set up the two multivalued dependencies (MVDs) needed.

Six MULTI-ASSOCIATION clauses appear in the definition, each needed to generate a separate MVD.

Despite the presence of separate MULTI-ASSOCIATION clauses for PAST-EMPLOYER and POSITION-HELD entities, the last MULTI-ASSOCIATION clause, which names both those entities, is required. This is because EMPLOYEE must be associated with properly matched pairs of instances of PAST-EMPLOYER and POSITION-HELD. The dependencies:

```
EMPL-NO ---->> EMPLR-NAME
EMPL-NO ---->> JOB-CATEG
```

are not equivalent in meaning to the single dependency:

```
EMPL-NO ---->> EMPLR-NAME, JOB-CATEG
```

6 Syntax of Entity Member Definition Statements

```
ENTITY
[IDENTIFIER IS data-element-list]...
[ONE-ATTRIBUTES ARE data-element-list]...
[MULTI-ATTRIBUTES ARE data-element-list]...
[ONE-ASSOCIATIONS TO entity-list]...
[MULTI-ASSOCIATION TO entity-list]...
[SUB-ENTITIES ARE entity-list]...
[common clauses]...
{;}
{.}
```

where:

data-element-list is:

data-element-name [, *data-element-name*] . . .

where *data-element-name* is the name of an ITEM or GROUP member.

entity-list is:

entity-name [, *entity-name*] . . .

where *entity-name* is the name of an ENTITY member .

common clauses are any of these clauses:

```
    ACCESS-AUTHORITY 'name' { [ READ-ONLY  }                    ]
                            { UPDATE       }
                            { SECURITY-CONTROL }
[,'name' [ {READ-ONLY       } ]]...
           {UPDATE          }
           {SECURITY-CONTROL}

    ADMINISTRATIVE-DATA 'text'
    ALIAS [alias-type] 'alias' [, [alias-type] 'alias']...
    CATALOGUE 'classification' [,'classification']...
    COMMENT 'text'
    DESCRIPTION 'text'
    EFFECTIVE-DATE date
    FREQUENCY { frequency                                                    }
              { { ACCESS  }      frequency [ ACCESS  ]  frequency]...       }
              { { RUN     }                { RUN     }                      }
              { { UPDATE  }                { UPDATE  }                      }
              { { BACK-UP }                { BACK-UP }                      }
```

where *frequency* is:

```
{ { MONTHLY } { [ number ] }
{ { WEEKLY  } { [ 'string' ] }
{ { DAILY   }
{ { HOURLY  }
{ number
{ 'string'
```

```
NOTE 'text'

OBSOLETE-DATE  date

QUERY 'text'

SECURITY-CLASSIFICATION 'string' [ FROM date ]
    [, 'string' [ FROM date ]]...

SEE member-name [ FOR 'string' ] [, member-name [ FOR 'string' ]]
```

Index